P.Texto(01).dot

MÓDULO

PROGRAMACIÓN BÁSICA EN TIA PORTAL CON S7-1200

UNIDAD 7

BLOQUES DE PROGRAMA

UD000100_V(01)

Instituto Tecnológico

ÍNDICE

MOTIVACIÓN 3
PROPÓSITOS 4
PREPARACIÓN PARA LA UNIDAD 5
1. ESTRUCTURA DE UN PROGRAMA 7
1.1. INTRODUCCIÓN 7
1.2. PROGRAMACIÓN ESTRUCTURADA. BLOQUES DE PROGRAMA. 9
1.2.1. EL BLOQUE FUNCIONAL 9
2. BLOQUES DE PROGRAMACIÓN 13
2.1. BLOQUES DE ORGANIZACIÓN (OB) 16
2.2. BLOQUES DE FUNCIÓN (FB) 18
2.3. BLOQUES DE FUNCIÓN (FC) 18
2.4. BLOQUES DE DATOS GLOBALES (DB) 18
2.5. LLAMADA A LOS BLOQUES 22
CONCLUSIONES 23
RECAPITULACIÓN 24
AUTOCOMPROBACIÓN 25
SOLUCIONARIO 29
PROPUESTAS DE AMPLIACIÓN 30
BIBLIOGRAFÍA 31

MOTIVACIÓN

Estructura. Esta es la palabra clave con la que vamos a trabajar en esta unidad didáctica. Un programa que ha de controlar un automatismo es un conjunto de elementos o condiciones que, mediante un adecuado orden, lleva a cabo un trabajo o unas operaciones predefinidas o previsibles.

Cualquier conjunto ordenado está formado por componentes que forman parte de ese conjunto. Pero no están de una forma cualquiera. Para que ese conjunto sea eficaz en su labor, ha de tener una estructura que permita una relación óptima y útil al fin para el que fue creado.

De la misma manera que un montón de ladrillos no son nada y con la adecuada estructura se convierten en un edificio, aprenderemos a estructurar nuestro programa más allá de realizar un listado infinito de instrucciones.

¿Consideras que es importante la estructura de un programa? Esperamos que tu respuesta sea “Sí”. Si no es así, estamos seguros de que después de haber estudiado esta unidad didáctica tu respuesta cambiará.

PROPÓSITOS

En esta unidad, aprenderemos a trabajar con bloques:

- Cómo estructurar nuestro programa mediante bloques.
- Crear bloques.
- Controlar el acceso a los bloques.
- Tipos de bloques.

PREPARACIÓN PARA LA UNIDAD

En esta unidad didáctica, no vamos a presentar un gran número de funciones con una aplicación específica, sino a ampliar el concepto que tenemos de programa.

Los conocimientos que adquirirás en esta unidad son de aplicación en cualquier programa que realices, pues afectan a la estructura misma de nuestra programación, es decir, al instante inicial en el que nos deberás plantear la forma en que organices la lógica del automatismo.

1. ESTRUCTURA DE UN PROGRAMA

1.1. INTRODUCCIÓN

En la elaboración de un programa y su posterior evaluación, hemos de tener en cuenta diversos aspectos, íntimamente relacionados con la programación.

Para elaborar un programa que responda a las necesidades de una automatización es muy importante:

- **Conocer qué es lo que queremos accionar.** Es decir, hemos de saber si hemos de mover un motor, accionar una electroválvula, etc., para, de esta forma, seleccionar el número de salidas que vamos a necesitar.

Por supuesto que habría que tener en cuenta más consideraciones, como tensiones de trabajo, potencias... Pero ahora nos vamos a centrar en la programación.

- **Conocer de qué información disponemos.** Es decir, saber de cuántas entradas disponemos. Entradas que nos darán la información del estado del sistema a automatizar.

En muchos casos, el programador es quien decide los elementos necesarios para tener la información suficiente procedente del sistema.

Dependiendo del grado de implicación del programador en la automatización, tendrá mayor control sobre el número de entradas y salidas del que dispone.

Si, además, el programador es quien construye la máquina, decidirá tanto las entradas como las salidas necesarias para que la máquina evolucione conforme a los requerimientos del cliente.

Si el programador tiene que hacer una modificación sobre una máquina o instalación cuyos actuadores (motores, válvulas…) ya están definidos, tendrá libertad para añadir alguna entrada que le facilite información de cara al nuevo programa.

Si el programador no interviene en la construcción de la máquina o instalación, recibirá un listado de entradas y salidas, y deberá realizar un programa para accionar las salidas indicadas en la secuencia requerida y con la información de entradas que se le aporta.

- Una vez que conocemos la información relativa a entradas y salidas, el siguiente pilar que conforma nuestro programa será **el conjunto de operaciones o instrucciones disponibles** en el *software* de programación y **comprensibles** por el autómata que vamos a utilizar para crear dicho programa.

Aparentemente, ya estarían determinados los componentes fundamentales para elaborar nuestro programa. Sin embargo, "no se vayan todavía, aún hay más".

- Otro pilar fundamental es la **estructura del programa.** Tan importante como el juego de operaciones disponibles en el autómata, resulta ser la forma en que organizamos todas esas instrucciones en dicho programa.
- El último elemento importante será la correcta **documentación** de todo el proyecto, que permitirá una rápida y eficaz depuración, seguimiento y posterior modificación.

1.2. PROGRAMACIÓN ESTRUCTURADA. BLOQUES DE PROGRAMA.

Figura 1. Estructura

En la realización de cualquier proyecto, sea la disciplina que sea, existe un elemento o concepto presente en todos ellos y que conocemos como **estructura.**

Si atendemos a la definición del diccionario, la estructura es "la disposición y orden de las partes dentro de un todo". También puede entenderse como un sistema de conceptos coherentes enlazados, cuyo objetivo es precisar la esencia del objeto de estudio. Tanto la realidad como el lenguaje tienen estructura.

Como no podía ser de otra manera, hemos de aplicar a nuestro programa un criterio estructural. Es decir, hemos de estructurar nuestra programación.

1.2.1. EL BLOQUE FUNCIONAL

El bloque funcional es un conjunto de elementos del programa, es decir, de líneas de programa, que tienen una característica común en cuanto a su función dentro de dicho programa.

El bloque funcional es una división del programa. Lógicamente, está íntimamente ligado al mismo, pues forma parte de él, pero nos sirve para poder seccionarlo cuando este se hace extenso.

La división en bloques de programa permite seccionar nuestro programa para hacerlo más manejable. Imaginemos un programa con un listado de cientos de líneas programadas. Resultaría muy complicado poder supervisar dicho programa, pues estaríamos sin ninguna referencia y, al final, todas las líneas de programa parecerían iguales.

Algo parecido a lo que ocurriría en un libro si no pudiéramos tener la referencia del índice.

Sin embargo, la división en bloques es algo más dinámica que el índice de un libro.

Al dividir en bloques estamos creando una jerarquía, una estructura que compondrá nuestro programa. Así, frente a la **disposición lineal** que hemos visto hasta ahora, en la que todas las instrucciones de programa estaban en el mismo bloque de programación, denominado principal, surgirán bloques "especializados" en los que se ejecutará una parte del programa.

El bloque principal es el eje del programa. Es donde el autómata lee las instrucciones que componen dicho programa.

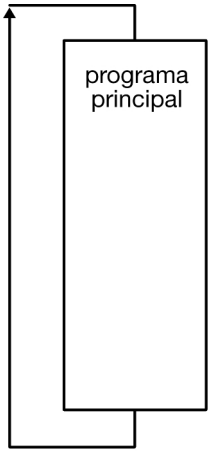

Figura 2. Programa principal

Lo que hace el autómata es leer todo lo contenido en un bloque de programa, al que se denomina **OB1.**

El autómata, en su ciclo, lee todas y cada una de las instrucciones contenidas en el bloque principal, es decir, dicho bloque es el lugar de la memoria del autómata donde este acude a leer las instrucciones que debe ejecutar.

Esto nos podría dar una idea excesivamente monolítica de un programa. Sin embargo, en cuanto un programa es mínimamente extenso se hace necesario dividirlo en subprogramas, creando así una estructura que nos aporte claridad y otras ventajas que veremos más adelante.

Es posible seccionar nuestro programa en pequeños subprogramas, los cuales estarán unidos entre sí o al bloque principal manteniendo una rígida jerarquía, tal y como podemos ver en la figura.

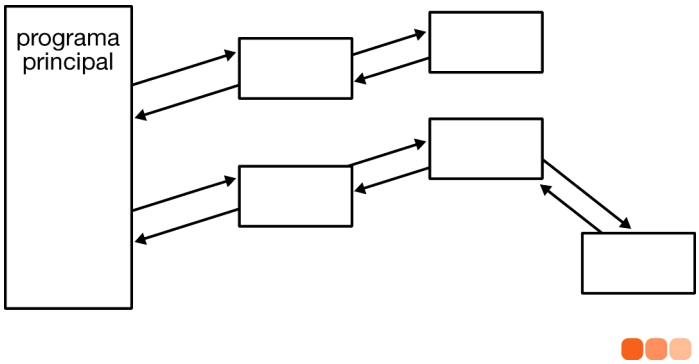

Figura 3. Programa principal y subprogramas

Como vemos en la figura superior, podemos generar una estructura de bloques que sale del bloque principal de forma que el programa, o, mejor dicho, la lectura del programa, viaja hacia ellos o desde ellos.

Realiza el siguiente ejercicio.

La _____[1] es la disposición y orden de un todo.

El bloque _____[2] es el eje del programa, donde el autómata lee las instrucciones que componen dicho programa. Es posible seccionar nuestro programa en pequeños _____[3], los cuales estarán unidos entre sí o al bloque principal.

Solución:
[1] Estructura.
[2] OB1.
[3] Subprogramas.

2. BLOQUES DE PROGRAMACIÓN

Para la programación en TIA-Portal podemos hacer uso de 4 tipos de bloques:

- Bloques de organización (OB)
- Bloques de función (FB)
- Funciones (FC)
- Bloques de datos (DB)

Podemos seleccionar cualquiera de ellos en el Árbol de Proyecto, dentro de nuestro PLC en la carpeta “Bloques de programa”

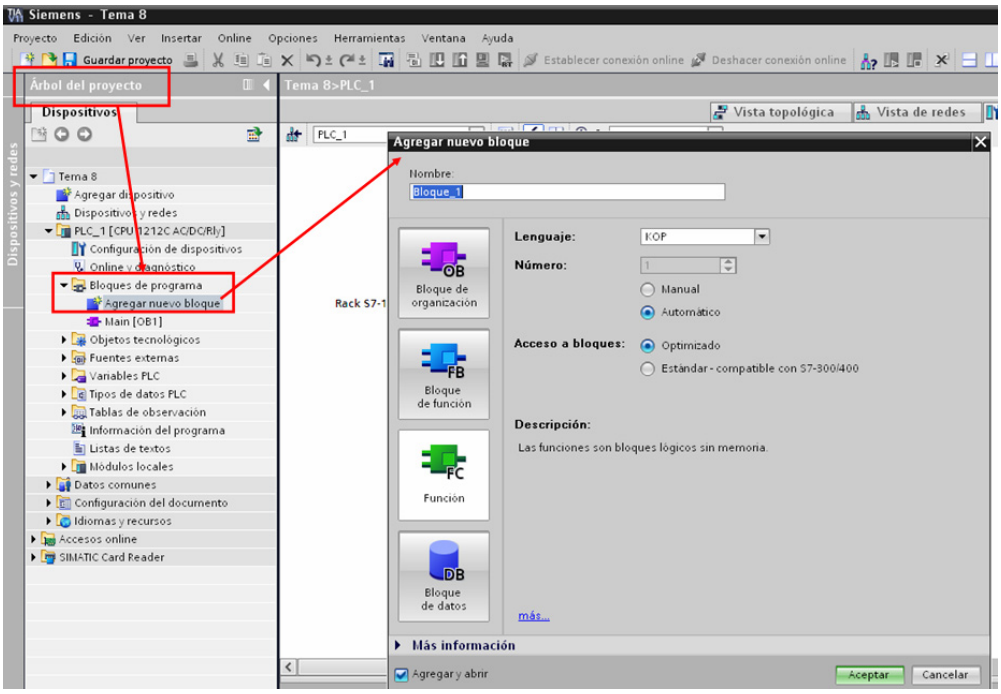

Figura 4. Cómo agregar un nuevo bloque

Una vez seleccionado el tipo de bloque y definidas sus características, simplemente deberemos darle a la tecla “Aceptar”.

Cuando creamos un nuevo bloque, este automáticamente nos aparecerá dentro de la carpeta de Bloques de programa. En la siguiente figura podemos ver los bloques de Prueba que hemos creado:

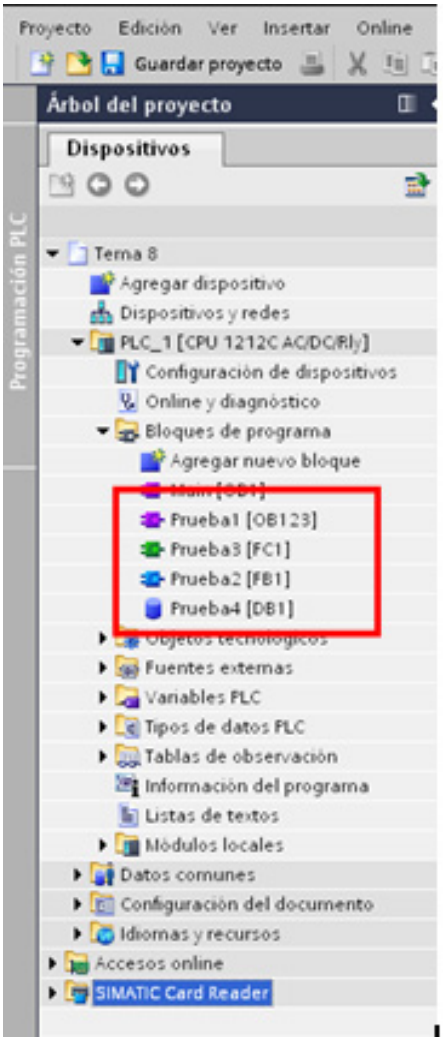

Figura 5. Bloques creados

Se puede acceder a todas las propiedades de estos bloques y modificarlas simplemente colocándonos sobre ellos en el Árbol de instrucciones, hacer clic con el botón derecho y seleccionar propiedades.

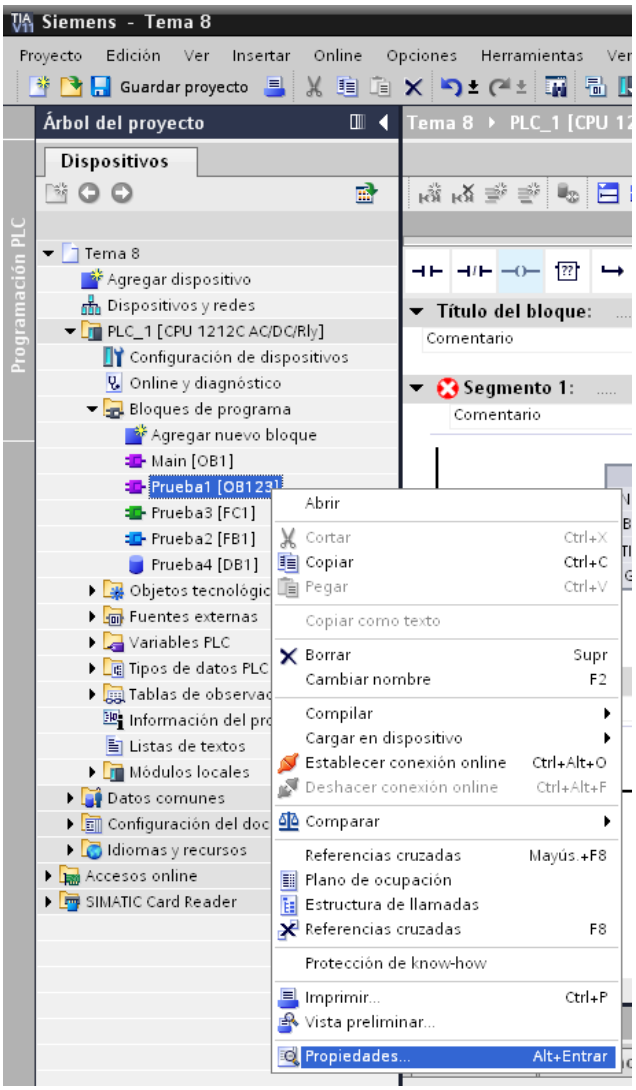

Figura 6. Acceso a las propiedades del bloque “Prueba1”

Veamos con más detalle cada uno de los tipos de bloques.

2.1. BLOQUES DE ORGANIZACIÓN (OB)

Son los bloques donde se realiza básicamente el programa.

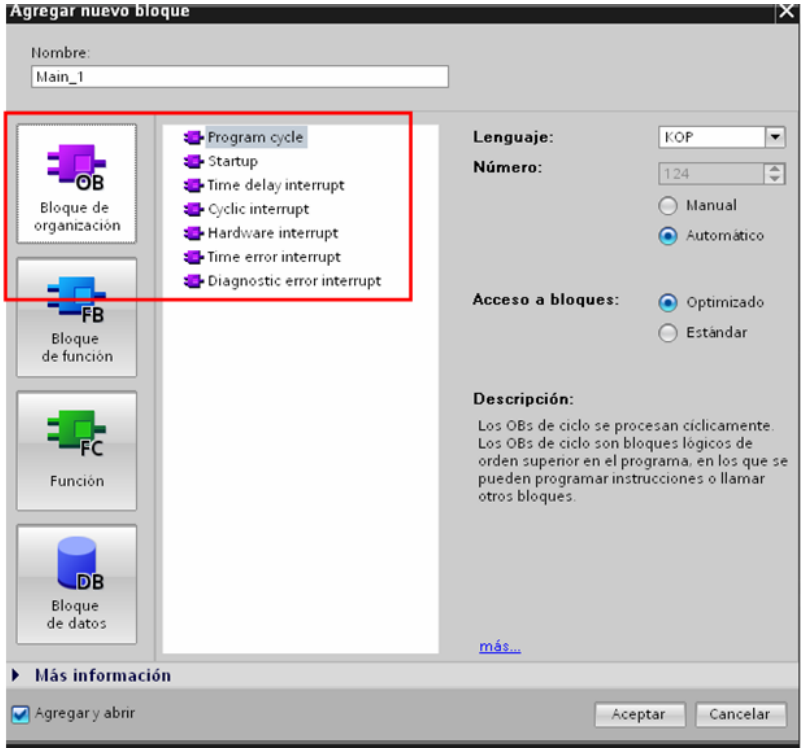

Figura 7. Selección de bloque OB

Existe siempre un bloque principal, llamado OB1, que se crea automáticamente al seleccionar la CPU. Este bloque se repite cíclicamente desde el momento del arranque y tiene la prioridad más baja de todos los OB, es decir su funcionamiento quedará interrumpido si otro OB es activado. Una vez finalizada la interrupción el sistema operativo rearranca el OB1.

En función de las necesidades del programador se pueden crear otros OB para estructurar mejor el programa.

Los bloques de organización son llamados por el sistema operativo, controlan principalmente los siguientes procesos:

- Comportamiento en arranque del sistema de automatización
- Ejecución cíclica del programa
- Ejecución del programa controlada por alarmas
- Tratamiento de errores

Programando los bloques de organización se define el comportamiento de la CPU. En función de la CPU utilizada, se dispone de diferentes bloques de organización. Veamos los principales tipos de OB:

- OB de ciclo o program cycle (OB>=123). Los OB de ciclo se procesan cíclicamente. En ellos que se pueden programar instrucciones o llamar otros bloques. Se pueden utilizar varios bloques de este tipo.
- OB de arranque o startup (OB100 ó OB>=123). Se ejecutan una sola vez cuando la CPU pasa de STOP a RUN. Cuando se ha ejecutado este bloque comienza la ejecución del OB de ciclo.

  Este tipo de bloque sirve para inicializar los valores del programa. No puede leer entradas ni activar salidas.
- OB de alarma de retardo o time delay interrupt (OB>=123). Interrumpen el procesamiento cíclico del programa transcurrido un tiempo definido. La llamada a este tipo de bloques se realiza a través de la instrucción avanzada "SRT_DINT" en la que se programa el tiempo de retardo.
- OB de alarma cíclica o cyclic interrupt (OB>=123). Interrumpen el procesamiento cíclico del programa en intervalos definidos. El valor de este intervalo se puede determinar cuando se crea el OB o posteriormente a través de sus propiedades.
- OB de alarma de proceso o hardware interrupt (OB<=123). Interrumpen el procesamiento cíclico del programa debido a un evento de hardware. El evento se define en las propiedades de hardware siguiendo los siguientes pasos:

  1. Haga doble clic en la entrada "Dispositivos y redes" del árbol del proyecto.
  2. El editor de hardware y redes se abre en la vista de red.
  3. Cambie a la vista de dispositivos.
  4. Si la ventana de inspección está cerrada en la vista de dispositivos, active la casilla de verificación "Ventana de inspección" del menú "Ver".
  5. Se abre la ventana de inspección.
  6. Haga clic en la ficha "Propiedades".
  7. En la vista de dispositivos, seleccione el módulo para el que desea parametrizar una alarma de proceso.
  8. Active el evento correspondiente.
  9. Introduzca un nombre de evento.
  10. Seleccione un OB de alarma de proceso existente de la lista desplegable "Alarma de proceso".

- OB de error de tiempo o time error interrupt (OB80). Interrumpe el procesamiento cíclico del programa cuando se rebasa el tiempo de ciclo máximo. El tiempo de ciclo máximo se define en las propiedades de la CPU.
- OB de alarma de diagnóstico o diagnostic error interrupt (OB82). Interrumpen el procesamiento cíclico del programa cuando el módulo apto para diagnóstico, para el que se ha habilitado la alarma de diagnóstico, detecta un error.

2.2. BLOQUES DE FUNCIÓN (FB)

Los bloques de función son bloques lógicos que depositan sus parámetros de entrada, salida y entrada/salida de forma permanente en bloques de datos de modo que siguen estando disponibles después de editar el bloque. Por eso también se denominan "bloques con memoria".

Los bloques de función contienen subprogramas que se ejecutan cada vez que un bloque de función es llamado por otro bloque lógico. Esto facilita la programación de funciones de uso frecuente.

2.3. BLOQUES DE FUNCIÓN (FC)

Las funciones FC bloques lógicos sin memoria. Las funciones son adecuadas para programar funciones complejas que se repiten con frecuencia (por ejemplo, cálculos).

Es importante que tengas en cuenta que los datos de las variables temporales se pierden tras haberse procesado la función. Si se quisieren almacenar los datos de forma permanente habría que configurar un bloque de datos para ello.

2.4. BLOQUES DE DATOS GLOBALES (DB)

Los bloques de datos sirven para almacenar datos de programa. Así, los bloques de datos contienen datos variables con los que trabaja el programa de usuario.

El tamaño máximo de los bloques de datos varía en función de la CPU.

Un bloque de datos no contiene ni segmentos ni instrucciones. Las declaraciones de variables definen la estructura del bloque de datos. Estos datos se conservan en el bloque de datos incluso al cerrarlo.

Es decir, un bloque de datos es como un edificio y en cada planta hay una variable. Así cuando queramos acceder a alguna de las variables de un DB, deberemos de utilizar el nombre del DB.nombre_variable

Por ejemplo,

En el bloque de datos Prueba4 que tenemos hemos creado una variable llamada "Rojo" con formato Word:

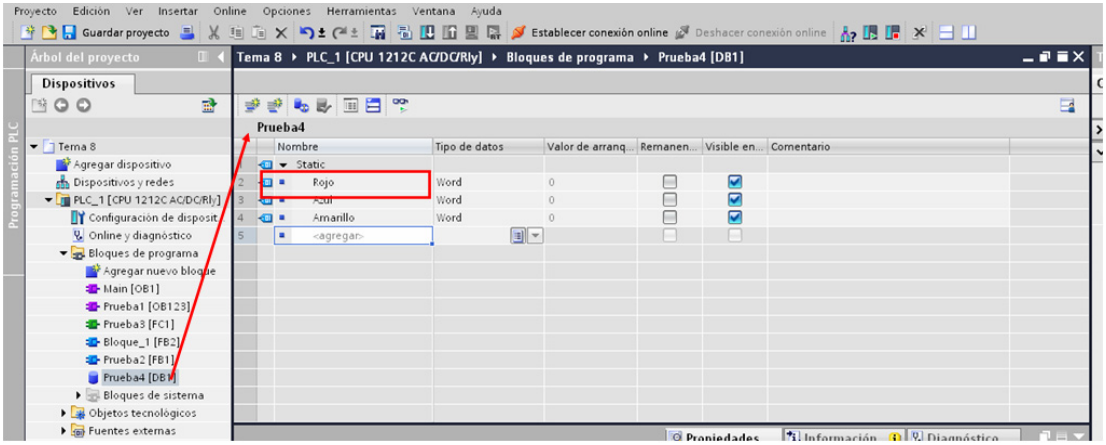

Figura 8. Configuración interna del DB Prueba4

Si quisiéramos utilizar esta variable, por ejemplo, en una comparación, la forma de utilizarla sería:

1. Insertamos la operación de comparación y seleccionamos el formato Word.

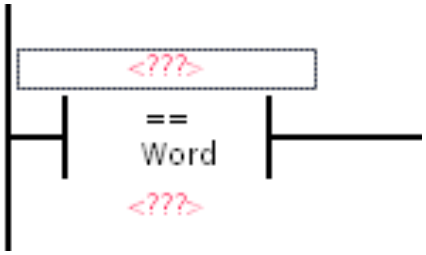

Figura 9. Selección operación de comparación

2. Empezamos a escribir el bloque de datos en la parte superior de la comparación:

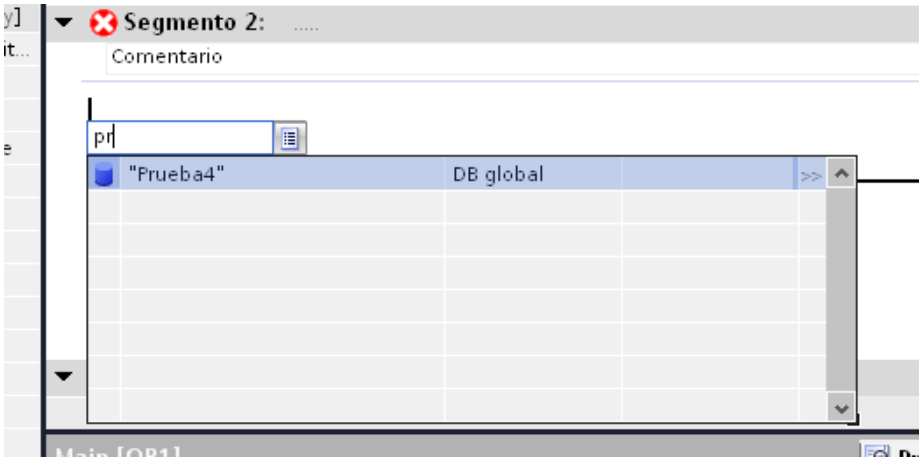

Figura 10. Selección del bloque de datos

3. Al seleccionar el DB nos aparecerán las variables internas y seleccionaremos la que queramos utilizar. En nuestro caso la variable “Rojo”.

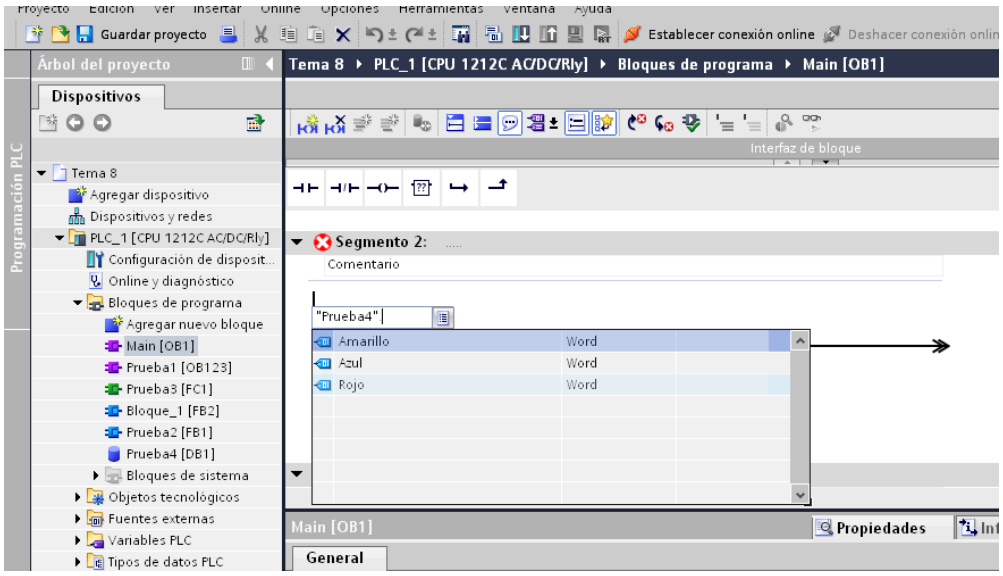

Figura 11. Selección de la variable

4. La variable ya queda insertada en la instrucción:

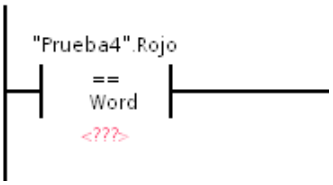

Figura 12. Variable “Rojo” seleccionada

Existen dos tipos de bloques de datos

- Bloques de datos globales

  Este tipo de bloque no está asignado a ningún bloque lógico. Se puede acceder a los valores de un bloque de datos global desde cualquier bloque lógico. Un bloque de datos global contiene únicamente variables estáticas.

  La estructura del bloque de datos global puede definirse libremente. En la tabla de declaración del bloque de datos se declaran los elementos de datos que debe contener el bloque de datos global.

- Bloques de datos de instancia

  Un bloque de datos de instancia es aquel que está asignado directamente a un bloque de función (FB). La estructura de un bloque de datos de instancia no puede definirse libremente, sino que está determinada por la declaración de interfaz del bloque de función. El bloque de datos de instancia contiene exactamente los parámetros de bloque y las variables declaradas en él.

  No obstante, en el bloque de datos de instancia pueden definirse valores específicos de la instancia, por ejemplo valores de arranque para las variables declaradas.

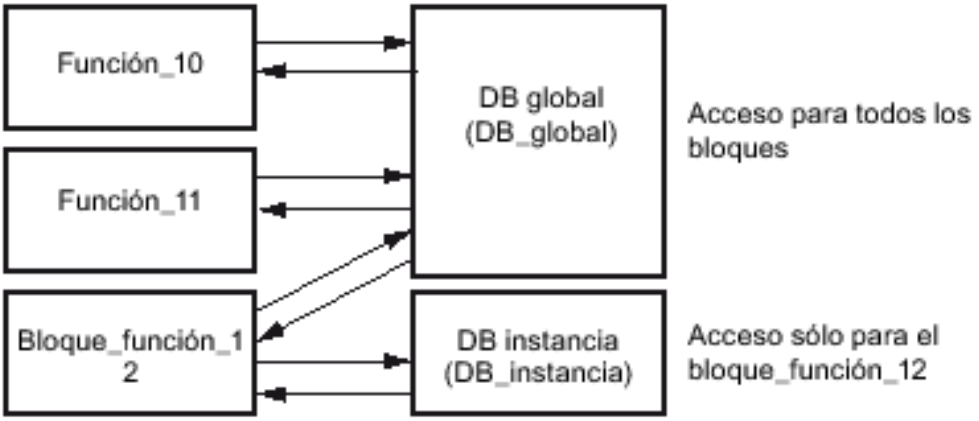

Figura 13. Acceso a los bloques de datos

Remanencia de los valores de datos

Para evitar pérdidas de datos en caso de un fallo de alimentación, los valores de datos pueden depositarse en un área de memoria remanente.

2.5. LLAMADA A LOS BLOQUES

Algunos de los bloques que hemos estudiado tienen ya configurada su propia llamada al asociarle un evento. Pero hay otros bloques que es necesario llamar desde el programa principal cuando queramos utilizarlos. Para ello simplemente arrastraremos el bloque desde el Árbol de proyecto hasta la línea de programa que queramos que active ese bloque: Insertaremos está función de la siguiente forma:

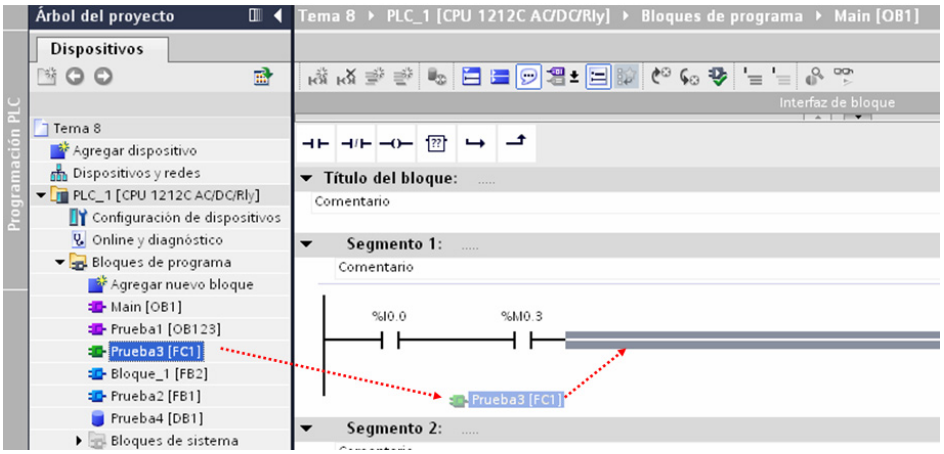

Figura 14. Arrastramos la función que queremos llamar

Soltamos el bloque al final de la línea:

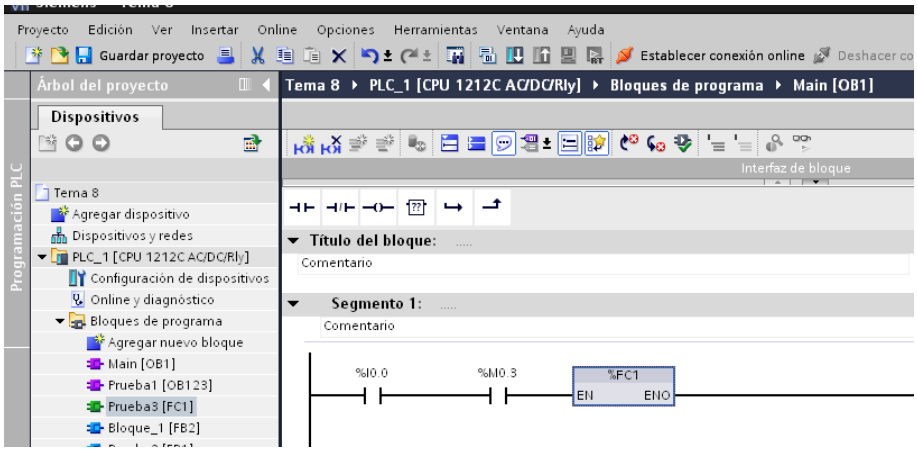

Figura 15. Llamada al bloque FC1

De esta forma podemos proceder con todos los bloques programables.

CONCLUSIONES

Ya hemos terminado la unidad didáctica. En ella hemos visto los tipos de bloques más importantes a la hora de estructurar un programa. Recuerda que mediante estas nuevas aplicaciones podemos realizar programas complejos pero que sean fácilmente entendibles.

Una vez que ya conoces los tipos de bloques, te proponemos que realices una tabla comparativa con las diferencias entre ellos.

RECAPITULACIÓN

En esta unidad, hemos conocido el uso de los bloques funcionales que vienen a complementar el bloque principal.

Hemos aprendido a trabajar con:

- OB
- Ccs
- FB
- DB
- Conocer la forma de llamar a estos bloques.

En definitiva, en esta unidad hemos aprendido a incorporar a nuestros programas nuevos bloques funcionales que nos permiten estructurar de forma más eficiente las distintas operaciones que deseamos que ejecute nuestro programa.

AUTOCOMPROBACIÓN

1. El OB principal es:

a) OB100.

b) OB1.

c) OB123.

d) OB80.

2. El OB100 se ejecuta al:

a) Siempre.

b) Nunca.

c) En el paso de Stop a Run del PLC.

d) Cuando se le hace la llamada desde el OB1.

3. Los Bloques de Organización son los:

a) FC.

b) FB.

c) OB.

d) DB.

4. Los Bloques de Función son los:

a) FB.

b) FC.

c) OB.

d) DB.

5. Los Bloques de Datos son los:

a) DB.

b) FB.

c) FC.

d) OB.

6. La llamada a un bloque:

a) Salta su ejecución.

b) No activa su ejecución.

c) Activan su ejecución.

d) Dependen del número de marcas.

7. Un bloque de datos de instancia es aquel que está asignado directamente a un:

a) DB.

b) OB.

c) FC.

d) FB.

8. Los bloques de datos sirven para:

a) Para generar segmentos.

b) Para reducir nuestro programa.

c) Deshabilita todos los eventos.

d) Almacenar datos de programa.

9. La división en bloques de programa permite:

a) Marca el orden en que se ejecuta.

b) Seccionar nuestro programa para hacerlo más manejable.

c) Ejecutar el programa más rápido.

d) Depende de la máquina.

10. Un bloque FC que no tiene la llamada:

a) Se ejecuta siempre.

b) Se ejecuta algunas veces.

c) No se ejecuta nunca.

d) Depende del autómata.

SOLUCIONARIO

1.	b	**2.**	c	**3.**	c	**4.**	a	**5.**	a
6.	c	**7.**	d	**8.**	d	**9.**	b	**10.**	c

PROPUESTAS DE AMPLIACIÓN

Te proponemos que identifiques tres procesos que puedas observar en tu entorno, en los que, por la rapidez o la precisión del mismo, se justificaría el uso de un OB configurado como interrupción. Piensa qué evento utilizarías.

BIBLIOGRAFÍA

- MANDADO PÉREZ, E. y otros. *Autómatas programables. Entorno y aplicaciones*. Madrid: Thomson-Paraninfo, 2005.
- ROLDÁN VILORA, J. *Automatismos y cuadros eléctricos*. Madrid: Thomson-Paraninfo, 1998.
- Documentación técnica del autómata S7-1200.
- Sistema de información de TIA Portal.